BIG

Revolutionizing how customers build, deploy and operate highly available and scalable services

Application Platform Infrastructure Team
Windows Server Product Group
Microsoft Corporation

June 19, 2002

Agenda

- Vision
- Architecture
  - *Hardware Reference Platform*
  - *Resource Management*
  - *IBIG*
  - *Service Definition Model (SDM)*
  - *Operations Logic*
- Customers and Scenarios
- Wrap-up

The Broad Vision

- **Enable**
  - ***development*** *of distributed, scalable and highly available services using Visual Studio and* ***reusable*** *building blocks like SQL, IIS, …*
  - ***deployment*** *across a set of* ***abstracted*** *hardware resources that are automatically allocated, purposed and configured*
  - ***lower cost of ownership*** *through automation by codifying operational best practices to control service availability and growth*
  - ***procurement*** *of standardized data center hardware that leverages* ***commodity economics***

BIG *is a revolutionary re-think of the way highly available and scalable services are built, deployed, operated*

BIG Will Deliver a Services Platform

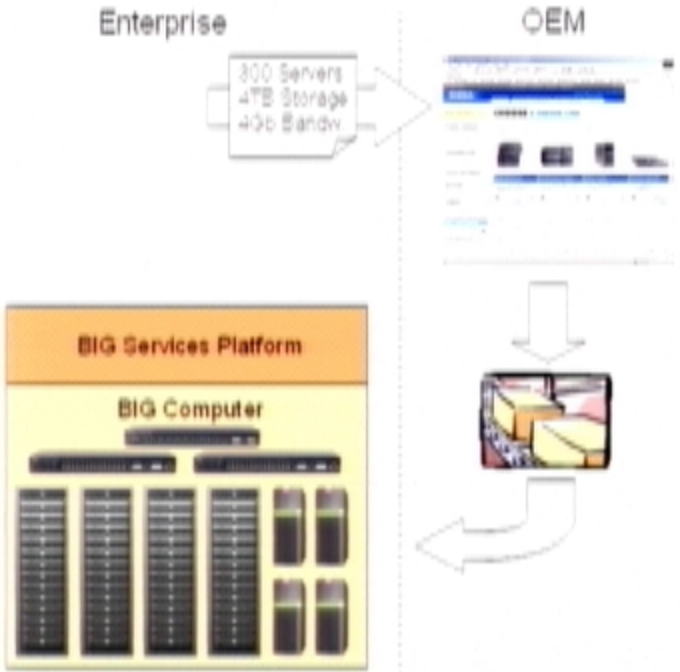
- **Hardware reference platform that aggregates commodity hardware to build a single large computer that we call the BIG Computer**
  - *Includes interconnected servers, network devices, and storage*
- **Hardware abstraction layer that virtualizes resources**
  - *Enables dynamic hardware binding and re-deployment and automated network configuration*
- **Service Definition Model (SDM) for developers to describe an entire service**
  - *Enables developers to rapidly build new services using highly available SQL, IIS and other reusable building block components*
  - *Requires incremental changes to BIG enable existing services*
- **Highly available runtime that supports the SDM**
  - *Enables hosting multiple scalable services inside the BIG Computer*
- **Operations logic framework for automating operational best practices**
  - *Enables policy expression and enforcement*

BIG Ecosystem

Enterprise

BIG Ecosystem

BIG Ecosystem



BIG Ecosystem

Developer

Enterprise

OEM

BIG Ecosystem

Developer

ASP.NET

Enterprise

OEM

BIG Ecosystem

Developer

ASP.NET

SQL

Enterprise

OEM

BIG Ecosystem

Developer

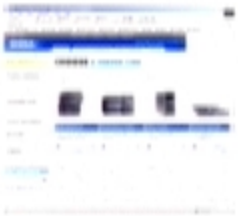
ASP.NET

SQL

Operations Logic

Enterprise

OEM



BIG Ecosystem

Developer

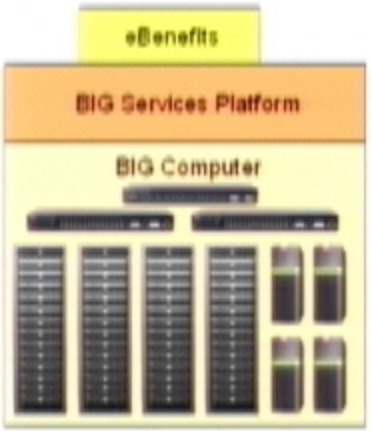
ASP.NET

SQL

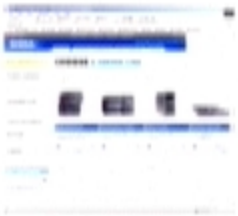
Operations Logic

Enterprise

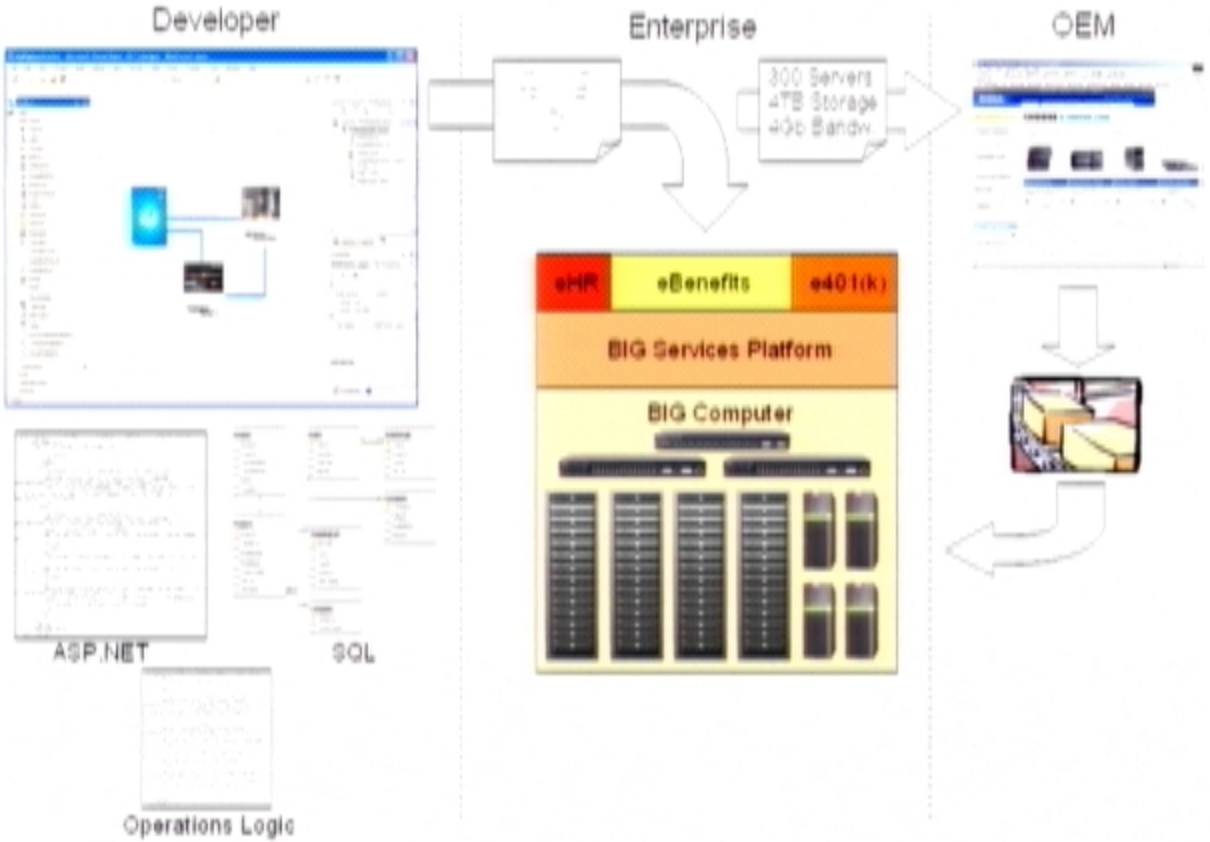
300 Servers
4TB Storage
40b Bandw



OEM



BIG Ecosystem

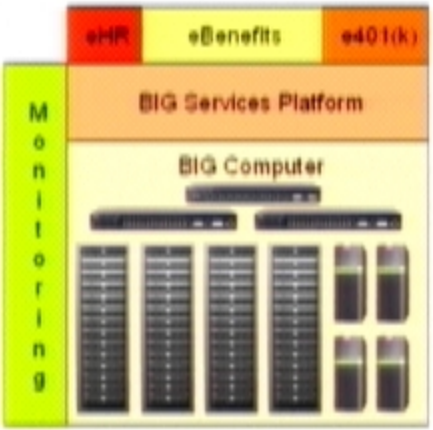


BIG Ecosystem

Developer

Enterprise

OEM

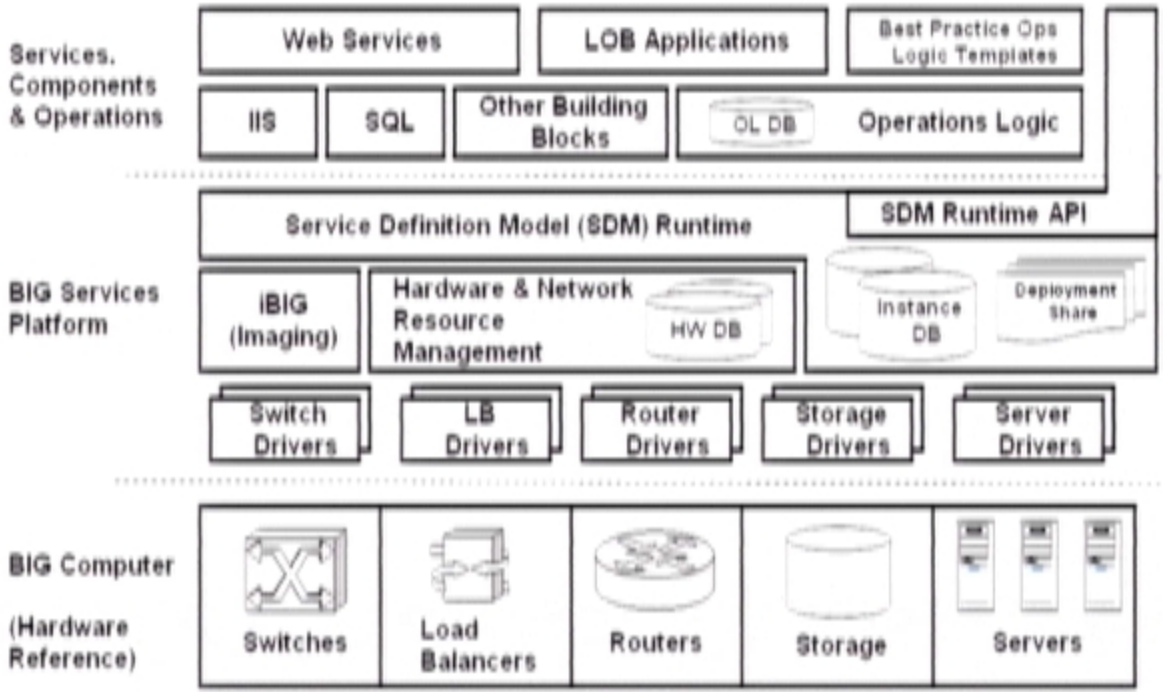


ASP.NET

SQL

Operations Logic

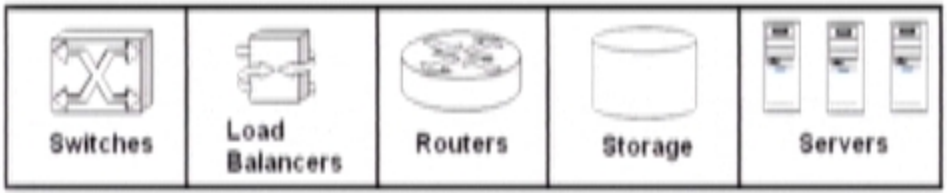
BIG Services Platform Architecture



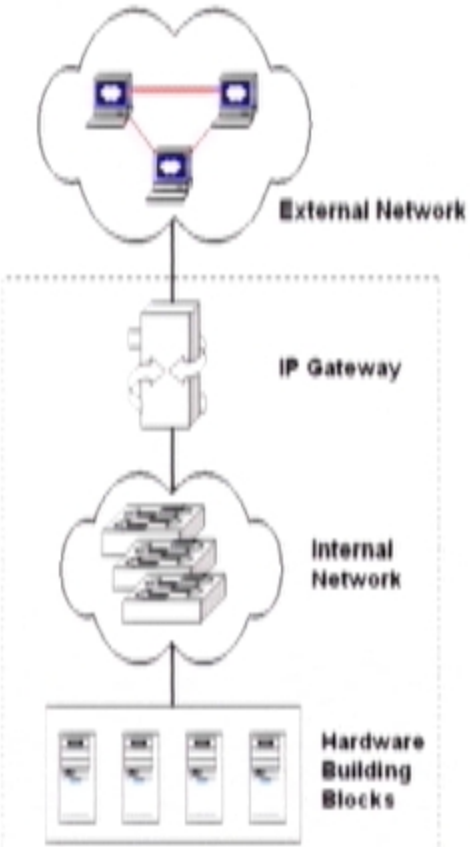
Hardware Reference Platform Initiative

BIG Computer

(Hardware Reference)



Hardware Reference Platform (BIG Computer)



Motivation

- Reduces the cost of design, test and operations
- Limits number of hardware devices to support
- Constrains the network topology and enables automation of network configuration
- Customers adverse to BIG taking over the entire data center (PXE, DHCP, DNS, network switches)

IP Gateway

- *Mediates IP traffic between the external network and the internal network*
- *Network Address Translation (NAT)*
- *ISA for firewall functionality*
- *Load Balancing*

Internal Network

- *IP addrs and VLANs are managed exclusively by BIG*
- *VLANs are automatically configured*

Hardware Building Blocks

- *Combinations of commodity servers, network switches, routers and disks*

Examples of Current Products that can be Inside a BIG Computer

BL10e

3U enclosure (20 blades)

7U enclosure (14 blades)

Compaq Proliant BL e-class

- Pentium III 700MHz
- 512MB – 1GB EC C RAM
- 30GB ATA Hard Disk
- Dual 10/100 Fast Ethernet
- Layer 2 switch with Gigabit uplinks
- Redundant 600 W power supplies
- 280 blades per 42U rack (2kW per slot)

IBM BladeCenter

- Dual Xeon
- 1GB EC C RAM
- 1 pci or Fibre Channel storage
- 4x Gigabit Ethernet
- 4x 1200 W power supplies
- 84 blades per 42U rack

Dell PowerEdge 1655MC

- Dual Pentium III 1.2GHz
- 1GB – 2GB EC C RAM
- 36-146 GB SCSI Hard Disk
- Dual Gigabit Ethernet
- 2 Layer 2 switches with Gigabit uplinks
- Redundant 1400 W power supplies
- 14 blades per 42U rack

3U enclosure (6 blades)

bh7800 rack (16 blades)

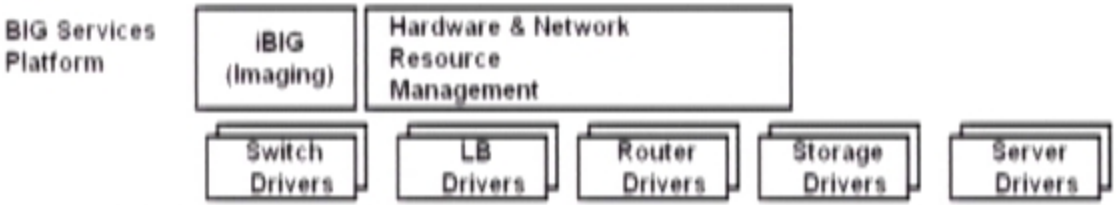
HP bc1100

- Pentium III 700MHz
- 512MB EC C RAM
- 30GB IDE Hard Disk
- Dual 10/100 Fast Ethernet
- 2 Layer 2 switches with Gigabit uplink
- Redundant 1200 W power supplies
- 80 W per slot
- 16 server blades per 19U rack

Fabric blades

- 1 Management blade
- 2 Management LAN blades
- 2 Network switch blades
- Storage blades (CE)

Infrastructure Services



Examples of Current Products that can be Inside a BIG Computer

BL10e

3U enclosure (20 blades)

7U enclosure (14 blades)

Compaq Proliant BL e-class

- Pentium III 700MHz
- 512MB – 1GB EDO RAM
- 30GB ATA Hard Disk
- Dual 10/100 Fast Ethernet
- Layer 2 switch with Gigabit uplinks
- Redundant 600 W power supplies
- 280 blades per 42U rack, 25W per slot

IBM BladeCenter

- Dual Xeon
- 2GB ECC RAM
- 1 SCSI or Fibre Channel storage
- 4x Gigabit Ethernet
- 4x 1200 W power supplies
- 84 blades per 42U rack

Dell PowerEdge 1655MC

- Dual Pentium III 1.2GHz
- 1GB – 2GB ECC RAM
- 36-146GB SCSI Hard Disk
- Dual Gigabit Ethernet
- 2x Layer 2 switches with Gigabit uplinks
- Redundant 1040 W power supplies
- 24 blades per 42U rack

3U enclosure (6 blades)

bh7800 rack (16 blades)

HP bc1100

- Pentium III 700MHz
- 512MB ECC RAM
- 30GB IDE Hard Disk
- Dual 10/100 Fast Ethernet
- 2x Layer 2 switches with Gigabit uplink
- Redundant 1200 W power supplies
- 80 W per slot
- 16 server blades per 19U rack

Fabric blades

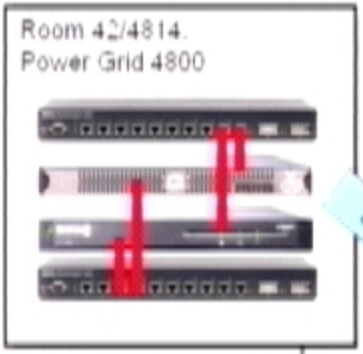
- 1 Management blade
- 2 Management LAN blades
- 2 Network switch blades
- Storage blades (CE)

BIG Resource Management

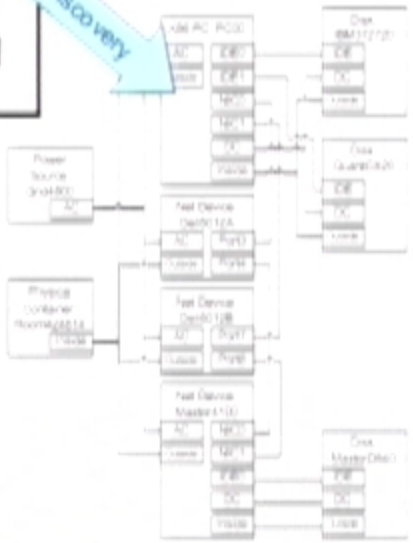
- **Global Resource Manager automatically allocates resources and binds services to the BIG Computer**
  - *Hardware devices are discovered and exposed as virtualized resources*
- **Resource providers**
  - *IIS VRoots*
  - *SQL Databases*
  - *CLR AppDomains*
  - *Win32 OS (iBIG for image deployment)*
  - *Network Resources*
  - *Bare metal PCs*
- **Resource Manager optimizes component placement to conserve valuable cross-sectional bandwidth and meet resource constraints**
- **Resource Manager frees operators from the detailed tasks of resource management**
  - *Automatically detects and updates system when new services or hardware are added to the BIG computer*
  - *Response to a failure event or policy action*

BIG Hardware Resource Discovery and Management

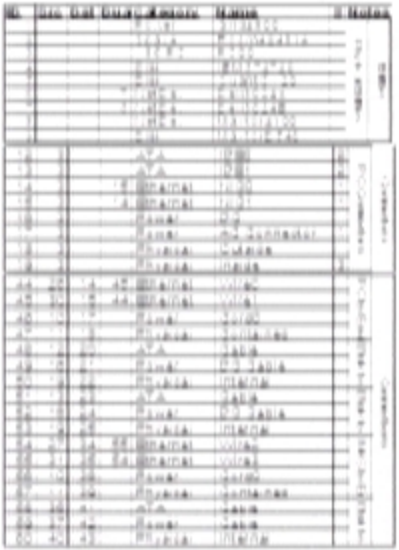
- Dynamic or static hardware discovery
- Resource drivers are bound to hardware devices and expose logical resources for allocation
- Allocation translates logical resource request to physical resource availability



Discovery



Physical Resource Graph



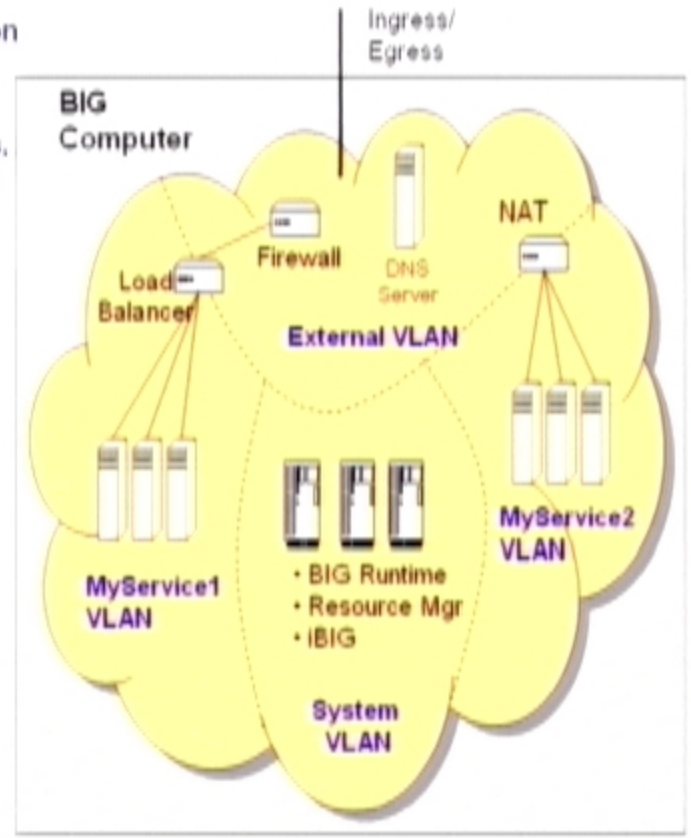
Physical Resource Database

Properties

- Power
- Network
- Storage
- Processor
- Memory
- Location

Network Management within the BIG Computer

- BIG Computer defines an abstraction layer for network resources
- BIG programs the network switches, load balancers and routers through software
- VLANs provide isolation
  - *Membership in a VLAN can be inferred from the BIG namespace*
- Network resources
  - *VLANs*
  - *Load Balancers*
  - *Firewalls*
  - *Routes*
  - *Network Filters*
  - *External IP addresses*
  - *External DNS Names*



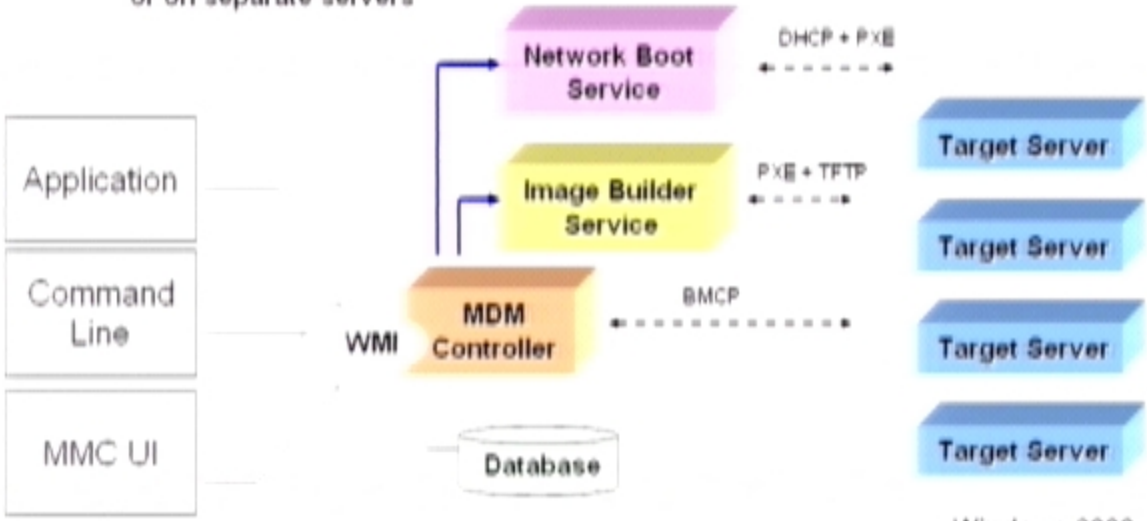
iBIG Features

- **Base Deployment Services**
  - *Basic Network Boot Service (PXE) and Image Builder Service*
  - *Pre-boot OS environment (BMonitor)*
  - *Virtual floppy delivered over network for legacy tools support*
- **Image Deployment and Management**
  - *Tools for creating, editing and deleting images*
  - *Deployment of images to systems running pre-OS*
- **Multiple Device Management (MDM)**
  - *Scripts for common tasks*
  - *Task sequencing to coordinate multiple steps and processes for deployment*
  - *Full programmatic interface (WMI)*
- **Supports Windows 2000 and .NET Server targets**

iBIG Architecture

Target servers run iBIG 'agent' to communicate with controller

Services can be co-located or on separate servers



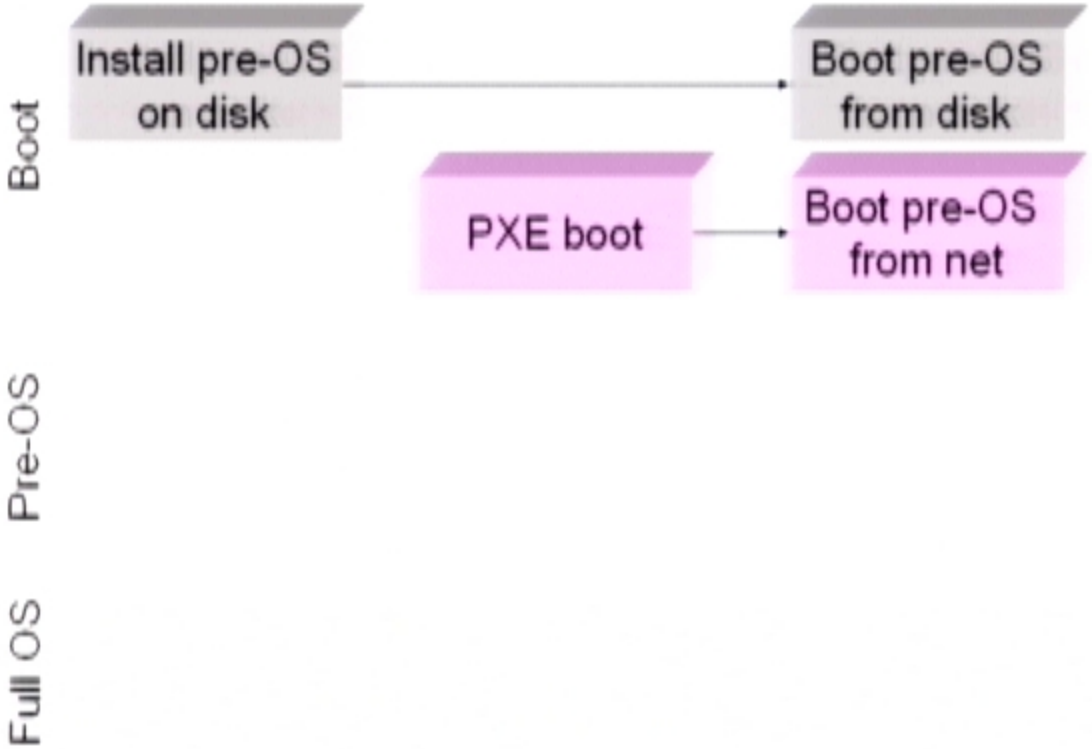
- Windows 2000
- .NET Server

- Controller runs only on .NET Enterprise Server
- WMI is the interface to the controller

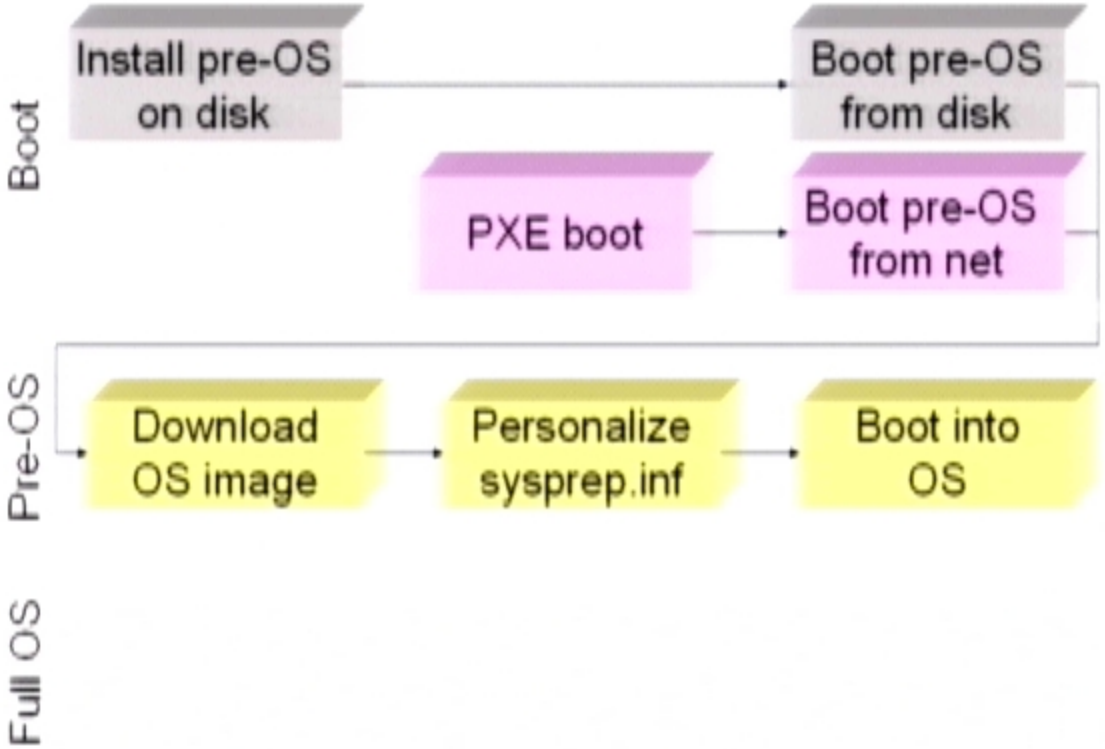
Full OS Pre-OS Boot

Server Purposing

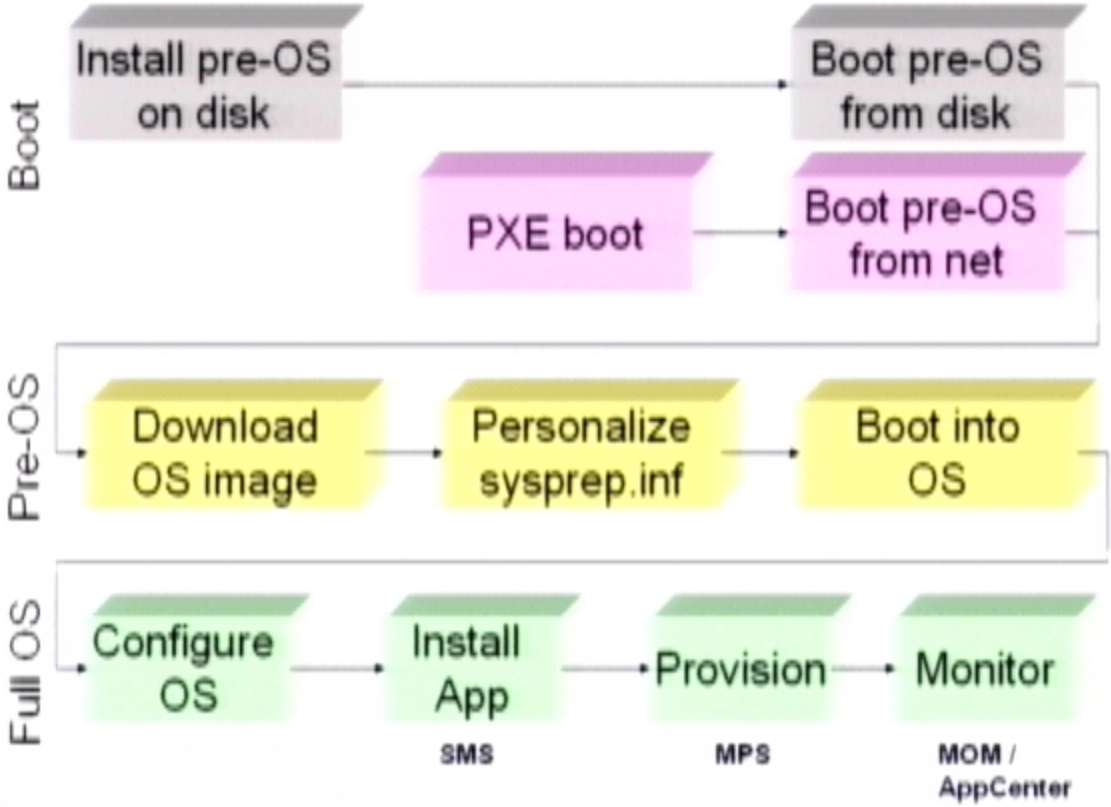
Server Purposing



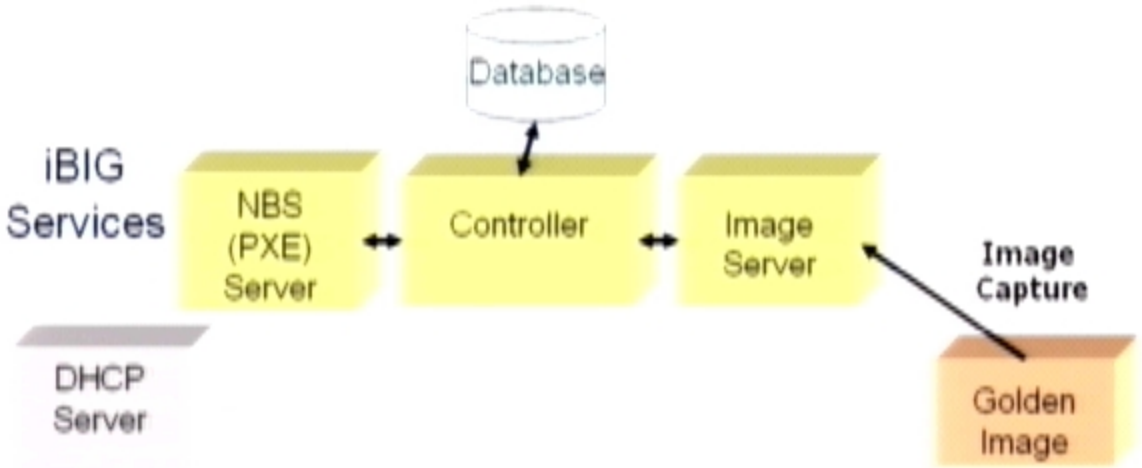
Server Purposing



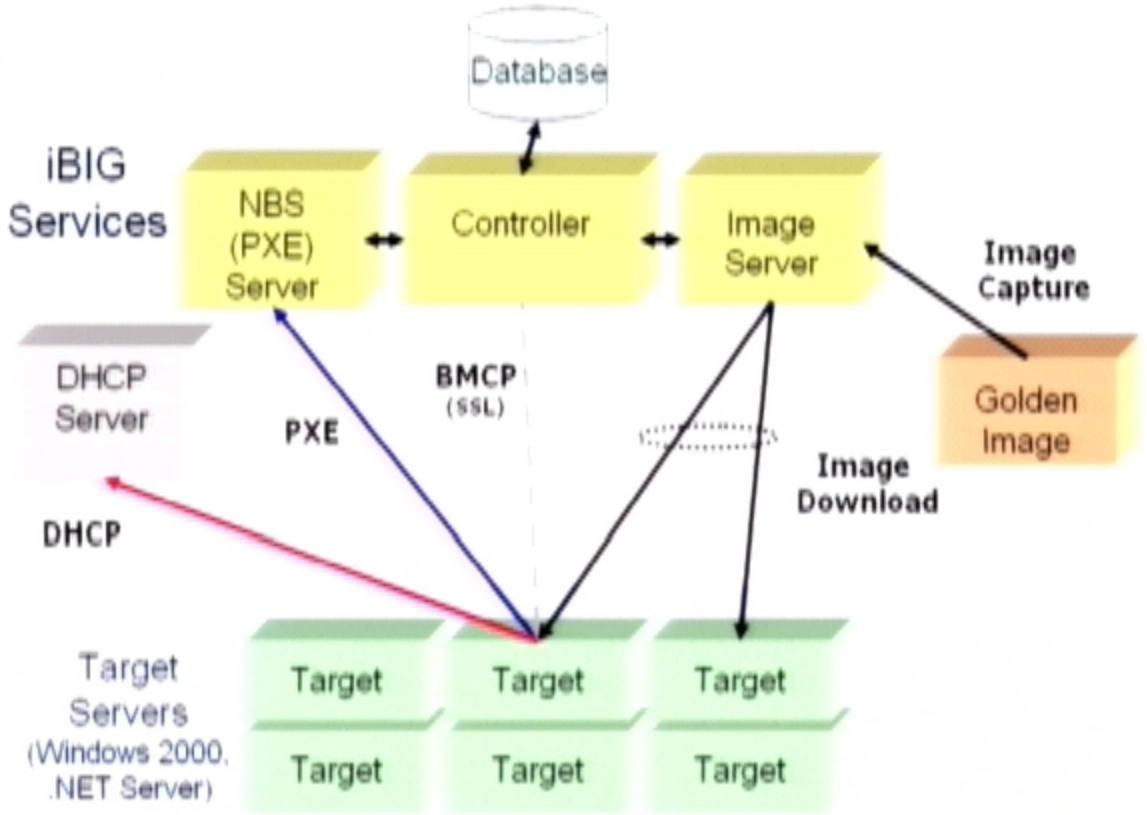
Server Purposing



Remote Boot and Imaging



Remote Boot and Imaging



Service Definition Model (SDM)

- **The programmatic description of the entire service**
  - ***Declarative*** *definition of the service*
  - *Defines the overall* **service structure** *of the service in a* **scale-invariant** *manner*
  - *Provides a* **framework** *for deployment, management, and operations*
  - ***Component-based*** *model captures in a modular fashion the elements of a service*
- **SDML is the declarative language for defining Service Definition Models**

Components, Ports and Wires

- **Components are units of implementation, deployment and operations**
  - *For example, dedicated server running .NET Server, IIS virtual web site, SQL database.*
  - *Expose functionality through ports and communicate through wires*
  - *Compound components created by composition*
- **Ports are names (service access points) with an associated type (protocol)**
  - *BIG does not mandate what protocols to use for communication*
  - *Protocols capture the information required for establishing communication*
- **Wires are the permissible bindings between ports**
  - *Wires declare a topological relationship between ports*

Components, Ports and Wires

- **Components are units of implementation, deployment and operations**
  - *For example, dedicated server running .NET Server, IIS virtual web site, SQL database, ...*
  - *Expose functionality through ports and communicate through wires*
  - *Compound components created by composition*
- **Ports are names (service access points) with an associated type (protocol)**
  - *BIG does not mandate what protocols to use for communication*
  - *Protocols capture the information required for establishing communication*
- **Wires are the permissible bindings between ports**
  - *Wires declare a topological relationship between ports*

SDML Example: MyService.sdml

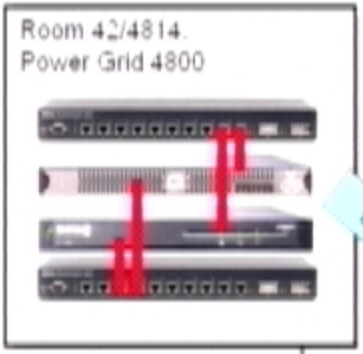
```
using System;
using System.SQL;
using System.IIS;
assembly name MyService;

componenttype MyFrontEnd :
   ASPApplication {
   port SQLClient catalog;
   implementation "MyFE, MyCLRApp"
}
componenttype MyBackEnd :
   SQLDatabase {
   implementation "MySQL, MyCLRApp"
}

componenttype MyService
{
   component MyFrontEnd fe;
   component MyBackEnd be;
   port http = fe.http;
   wire TDS tds {
         fe.catalog;
         be.sql;
   }
   implementation "MyService, MyCLRApp"
}
```

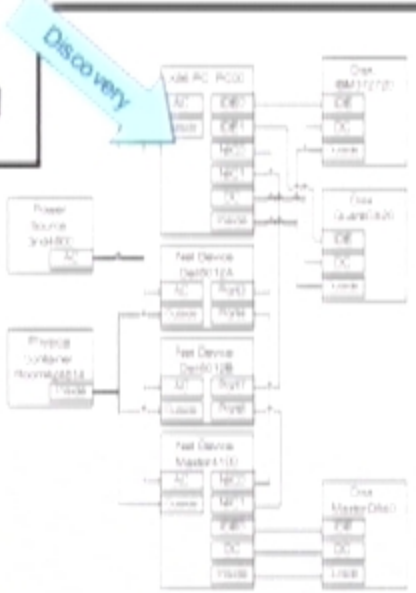
Components, Ports and Wires

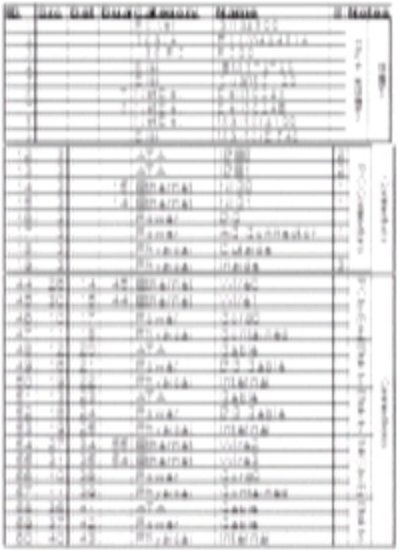
- **Components are units of implementation, deployment and operations**
  - *For example, dedicated server running .NET Server, IIS virtual web site, SQL database.*
  - *Expose functionality through ports and communicate through wires*
  - *Compound components created by composition*
- **Ports are names (service access points) with an associated type (protocol)**
  - *BIG does not mandate what protocols to use for communication*
  - *Protocols capture the information required for establishing communication*
- **Wires are the permissible bindings between ports**
  - *Wires declare a topological relationship between ports*

BIG Hardware Resource Discovery and Management

- Dynamic or static hardware discovery
- Resource drivers are bound to hardware devices and expose logical resources for allocation
- Allocation translates logical resource request to physical resource availability







Physical Resource Graph

Physical Resource Database

Properties

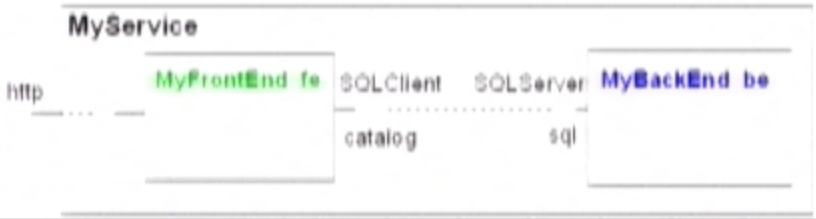
- Power
- Network
- Storage
- Processor
- Memory
- Location

Components, Ports and Wires

- **Components are units of implementation, deployment and operations**
  - *For example, dedicated server running .NET Server, IIS virtual web site, SQL database.*
  - *Expose functionality through ports and communicate through wires*
  - *Compound components created by composition*
- **Ports are names (service access points) with an associated type (protocol)**
  - *BIG does not mandate what protocols to use for communication*
  - *Protocols capture the information required for establishing communication*
- **Wires are the permissible bindings between ports**
  - *Wires declare a topological relationship between ports*

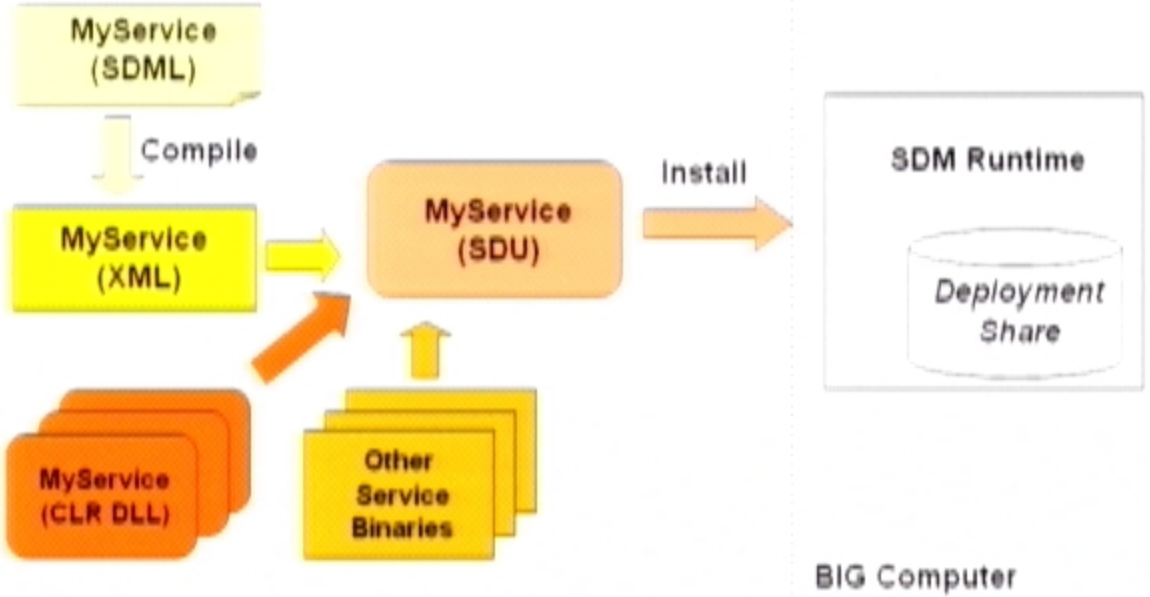
SDML Example: MyService.sdml

```
using System;
using System.SQL;
using System.IIS;
assembly name MyService;

componenttype MyFrontEnd :
   ASPApplication {
   port SQLClient catalog;
   implementation "MyFE, MyCLRApp"
}
componenttype MyBackEnd :
   SQLDatabase {
   implementation "MySQL, MyCLRApp"
}

componenttype MyService
{
   component MyFrontEnd fe;
   component MyBackEnd be;
   port http = fe.http;
   wire TDS tds {
         fe.catalog;
         be.sql;
   }
   implementation "MyService, MyCLRApp"
}
```



Service Deployment Unit (SDU)

- **Encapsulates all the pieces that make up a service, including**
  - *SDM model for the service*
  - *CLR assemblies for component implementation*
  - *MSI, ASP.NET, SQL scripts, Static content etc.*



SDM Runtime

- **SDM runtime is responsible for tracking SDM models and running instances**
  - *Implemented as a Web Service hosted by IIS*
  - *Exposes SOAP endpoints as SDM Runtime API*
  - *Can be partitioned for scalability*
- **Components perform operations using the SDM Runtime such as**
  - *Instantiating a component*
  - *Setting port information*
  - *Discovering peer component instances*
- **Highly available instance store (using Yukon's redundant database technology)**
  - *Two SQL servers and a witness server*
- **Security and account management using Active Directory**

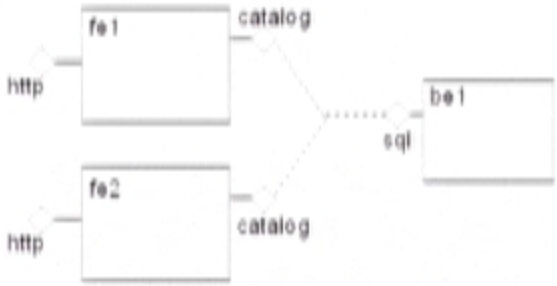
Example: Component Instantiation

```
using Microsoft.SDM;

public class MyService:
    SDMComponent
{
    public OnCreate(...)  {

    fe1 = CreateInstance("fe", "");
    fe2 = CreateInstance("fe", "");
    be1 = CreateInstance("be", "");

    w1 = CreateWireInstance("tds");
    w1.Members.Add(fe1.Ports["catalog"]);
    w1.Members.Add(fe2.Ports["catalog"]);
    w1.Members.Add(be1.Ports["sql"]);

    }
}
```

myservice.cs is C# code that uses the SDM API

```
componenttype MyService
{
    component MyFrontEnd fe:
    component MyBackEnd be:
    port http = fe.http:
    wire TDS tds (
            fe.catalog:
            be.sql:
    )
    implementation "MyService.
    MyCLRApp"
}
```

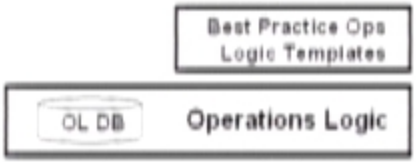
myservice.sdml



SDM Instances

Operations Logic

Services,
Components
& Operations



Operations Logic is the "Business Logic" of Operations

- **Operations Logic is CLR code that captures repeatable patterns encoded as reusable best practices**
  - *Not specific to a service or operating environment*
  - *Can be developed, tested and shipped*
  - *Reduces the need for manual procedures that require people to execute them*
- **OpsLogic is responsible for the overall operation of a service**
  - *Starting up a service*
  - *Service growth and shrinkage*
  - *Upgrades and updates*
  - *Fault detection and recovery*
  - *Database partitioning*
- **OpsLogic is implemented using MS middle-tier technologies**
  - *ASP.NET web services running on IIS*
  - *DTC for transaction coordination*
  - *SQL server for storage*
  - *WMI for monitoring and management*
  - *MSMQ for messaging*

Repeatable Upgrade Patterns → Operations Logic

- **Upgrade is an example of the type of Operations Logic we want to build and ship with BIG**

- **In-place Upgrade Pattern**
  - *Cost of moving data is high, code instantiation cost is low, or no spare resources*
  - *Takes component out of service, runs update, put it back in service*
- **Side-by-side Upgrade Pattern**
  - *Cost of moving data is low, code instantiation cost is high, have spare resources*
  - *Create new component; Take old component out of service; Migrate data to new component; Put new component into service*
- **Replacement Upgrade Pattern**
  - *No data migration*
  - *Add new components; remove old ones; coordinate to maintain service availability*

- **Rolling Upgrade is an example of higher-level operations logic that can reuse the codified upgrade patterns**
  - *Operations logic can be tested and the framework supports rollback*
  - *Removes human error from execution by letting software perform the steps*

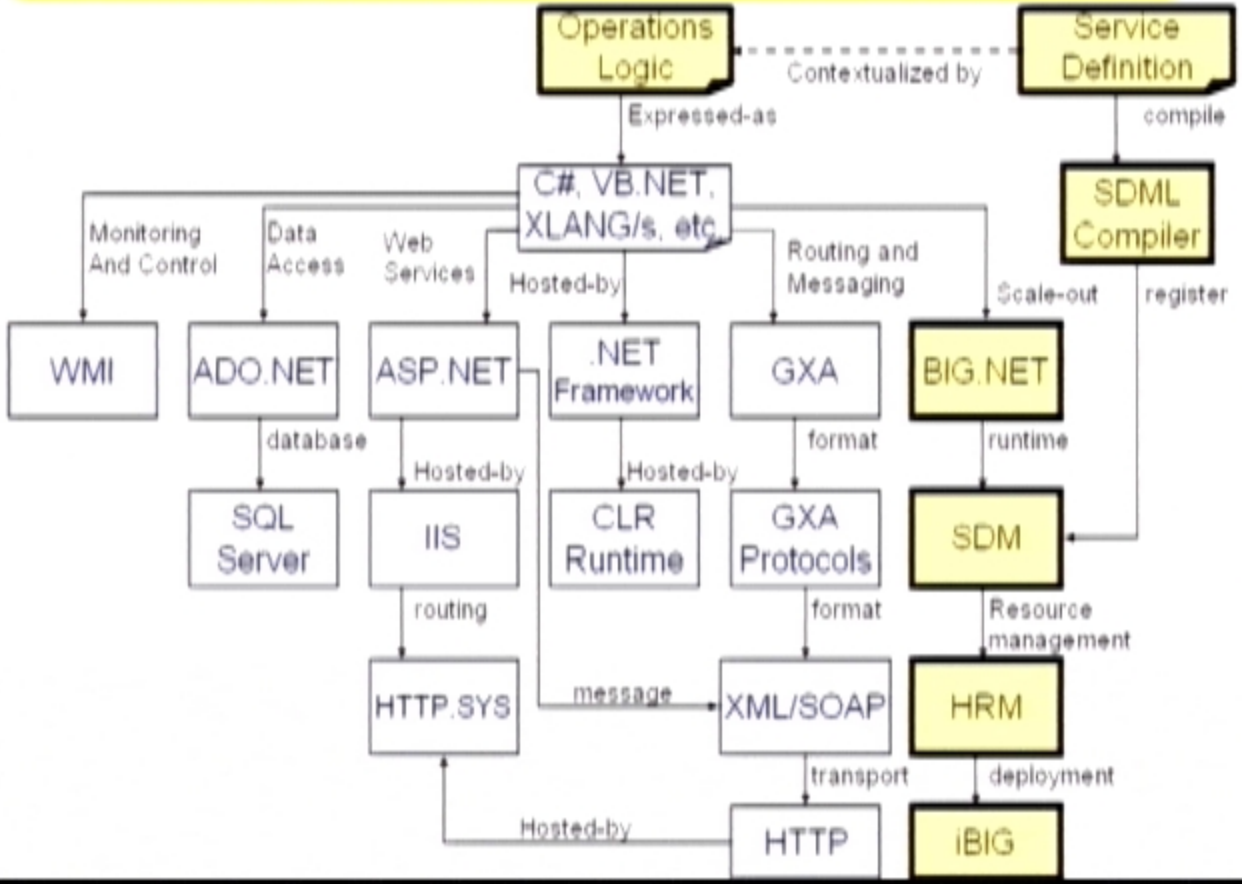
Repeatable Upgrade Patterns → Operations Logic

- **Upgrade is an example of the type of Operations Logic we want to build and ship with BIG**

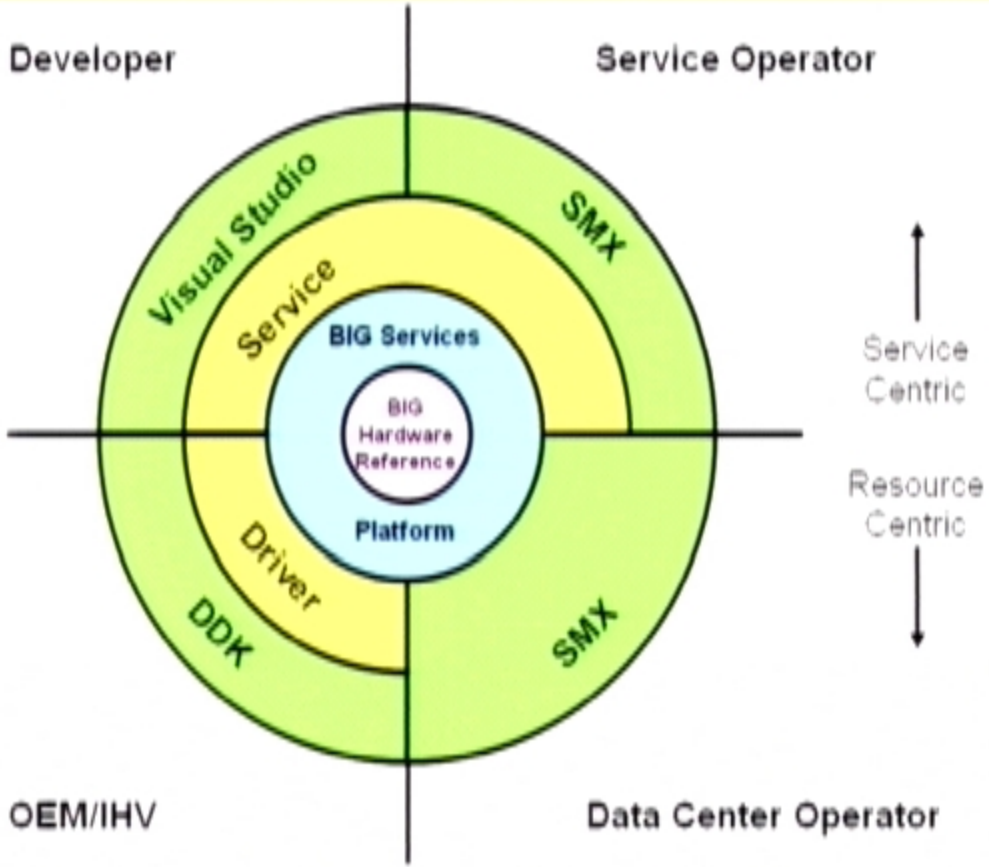
- **In-place Upgrade Pattern**
  - *Cost of moving data is high, code instantiation cost is low, or no spare resources*
  - *Takes component out of service, runs update, put it back in service*
- **Side-by-side Upgrade Pattern**
  - *Cost of moving data is low, code instantiation cost is high, have spare resources*
  - *Create new component; Take old component out of service; Migrate data to new component; Put new component into service*
- **Replacement Upgrade Pattern**
  - *No data migration*
  - *Add new components; remove old ones; coordinate to maintain service availability*

- **Rolling Upgrade is an example of higher-level operations logic that can reuse the codified upgrade patterns**
  - *Operations logic can be tested and the framework supports rollback*
  - *Removes human error from execution by letting software perform the steps*

Operations Logic, BIG and the Microsoft Programming Model



Customer View of BIG



Key Customer Scenarios

- **Enterprise computing utility**
  - *IT departments as corporate computing utilities to better utilize assets by separating LOB application deployment from hardware procurement*
  - *Example: Goldman Sachs*
- **Mega-services**
  - *Large-scale services that span multiple servers performing parallel or clone functions*
  - *Example: MSN Hotmail*
- **Shared-hosting**
  - *Multiple services hosted on a single server with or without a database*
  - *Example: Schlund or Digex*
- **Diskless network boot**
  - *Enables a single image to be loaded from the network and run in RAM that is not persisted across reboots to enable centralized image management*
  - *Enables diskless server blades (lower cost/power, increases reliability and density)*
  - *Example: Morgan Stanley*

Schedule

- **Timeline:**
  - *Current team focused on spec & design of product, building core system and integration with IBIG by end-2002.*
  - *Test team and additional dev to roll-off from IBIG at end of 2002.*
- **M2 – 6/28/02**
  - *BIG computer built in lab*
  - *Standalone resource manager and drivers running on BIG computer.*
  - *Standalone runtime and components (SQL, IIS) on single machine configuration.*
- **M3 – 8/30/02**
  - *Integrated network management, resource manager and runtime on BIG computer.*
  - *BIG applications using IIS and SQL running on BIG computer.*
- **M4 – 11/22/2002**
  - *Integration with IBIG to enable bare metal deployment*
  - *System recovery support*
- **RTM - Ship by end of 2003**

Summary

- **BIG proposes a hardware reference platform to lower cost of ownership**
  - *BIG Computer simplifies design, deployment and management of distributed, scalable and highly available services*
- **BIG is developer-centric, focused on extending the Windows Server Platform for building new services using Visual Studio and reusable building blocks like IIS, SQL, …**
  - *BIG provides technology to build, deploy and operate highly available and scalable services*
- **BIG abstracts away the hardware to enable dynamic binding and re-deployment and automated network configuration**
  - *BIG delivers infrastructure such as the SDM runtime, resource management, network boot, imaging and file distribution*
- **BIG schedule is targeting end of 2003**
  - *No dependency on Longhorn*